



UNITED STATES PATENT AND TRADEMARK OFFICE

mn

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/654,208	09/03/2003	Joseph R. Holman	MSFT120610	4630
26389 7590 08/03/2007 CHRISTENSEN, O'CONNOR, JOHNSON, KINDNESS, PLLC 1420 FIFTH AVENUE SUITE 2800 SEATTLE, WA 98101-2347			EXAMINER CHOW, CHIH CHING	
			ART UNIT 2191	PAPER NUMBER
			MAIL DATE 08/03/2007	DELIVERY MODE PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary

Application No.

10/654,208

Applicant(s)

HOLMAN ET AL.

Examiner

Chih-Ching Chow

Art Unit

2191

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 31 January 2007.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-49 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-49 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☒ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 22 December 2003 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- ☐ Notice of References Cited (PTO-892)
- ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____.
- ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date _____.
- ☐ Notice of Informal Patent Application
- ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to amendment dated January 31, 2007.
2. Per Applicants' request, none of the claims have been amended.
3. Claims 1-49 remain pending.

Response to Arguments

4. Applicants' arguments for Claims 1-49 have been fully considered respectfully by the examiner but they are not persuasive.
5. Applicants' arguments are basically in the following points:

Claim 1

- "Clearly, determining the integrity of a file (i.e., that it has not been corrupted), as purportedly disclosed by Ferchau, is not the functional equivalent to determining whether a software application is properly installed on a target computer."... "In regard to 'a validation manifest associated with the software application,' applicants assert that while the portable cryptographic device described in Ferchau may include a list of files to be checked for corruption, the device and its list of files are not associated with the software application.... Nothing in Ferchau discloses that the list of files on the portable cryptographic device bears an association with determining whether a software application is properly installed on a target computer" (See REMARKS dated 1/31/2007, pages 3 - 4) .

Examiner's Response: In response to applicant's argument that Ferchau's teaching determines the integrity of a file which is not the same as determining whether a software application is properly installed on a target computer – the argument is not persuasive, Ferchau's teaching validates the functions of the installed software of a cryptographic device, see Ferchau's paragraph [0038], "Security program 106 executes

on computing device 104 and contains program logic to communicate with a coupled portable cryptographic device 102 through connection 108 to **facilitate the software verification and other security functions.**”, further [0039], “security program 106 contains program logic to perform the validation and security Functions as described herein.”, and paragraph [0056], “The newly computed software verification value is compared to the software verification value received from security program 106. Network attached cryptographic device 110 sends the **comparison results to security program 106**, for example, over connection 112. Security program 106 then takes appropriate action depending on the received comparison results.” – In Ferchau’s teaching, the security program, when executed, provide results for determining whether or not a software application is validly installed on a target computer – it teaches all the aspects of the current application; Ferchau’s disclosure teaches even narrower scope of the validation function than the current application, therefore Ferchau’s teaching reads on current application.

Claims 2-12

- “Claim 2 recites that one of the validation actions in the validation manifest is the execution of ‘a validation program associated with the software application.’ In contrast, Ferchau purportedly discloses a security program that is entirely unrelated to the software application. Indeed, it would appear that this independence is maintained in order to ensure the integrity of the security program. Thus, Ferchau's security program is independent of the data files that it validates, without any disclosure of "a validation program associated with the **software application.**" .”(See REMARKS dated 1/31/2007, page 5)

Examiner’s Response: The argument is not persuasive, see Ferchau’s Abstract, “A method for facilitating the integrity validation of data files includes providing a portable cryptographic device **having a software verification key, and coupling the portable**

cryptographic device to a computing device. The method also includes **identifying a target data file for validation** on the computing device, and **generating a software verification value for the target data file using the software verification key**. An apparatus for facilitating the integrity validation of data files includes a software verification key that is provided on a portable cryptographic device. The apparatus also includes a security logic that is coupled to the software verification key” – the security program (security logic) is related to the software application, the security program is coupled to the software verification key, therefore it’s not completely independent from the software application.

Claim 3

- “Claim 3 recites that one of the validation actions in the validation manifest is the execution of ‘a validation routine in a loadable module associated with the software application.’ As already mentioned above, the security program on the portable cryptographic device is entirely independent and unrelated to the software application. Since Ferchau's security program is independent of the content that it validates, Ferchau clearly fails to disclose "a validation routine in a loadable module associated with the software application.” (See REMARKS dated 1/31/2007, page 6)

Examiner’s Response: The argument is not persuasive, see Ferchau’s paragraph [0011], “The present disclosure is directed to an apparatus and corresponding methods that detect unauthorized changes (e.g., modifications, substitutions, additions, deletions, etc.) to a data files, including software programs, such as, by way of example, **operating system software and application software**. The apparatus and methods may alert the authorized device user or administrator of unauthorized or improper changes to data files used in conjunction with computing devices, appliances, or other devices that utilize one or more data files.” and [0022], “A portable cryptographic apparatus and

corresponding methods, according to one embodiment, **facilitates the detection of unauthorized or improper changes to data files, including software (e.g., operating system software, application software, etc.), used in conjunction with computing devices, appliances, or other devices that require or use one or more data files.”**

– Ferchau’s disclosure teaches a loadable module associated with the installed software application, facilitates the detection of improper changes, therefore it has to be ‘associated’ with the software application.

Claims 5-10

- “Ferchau discloses using various aspects (with others) in generating a verification value for a particular file that is compared for equivalence to a previously generated verification value. Ferchau fails to disclose explicitly comparing certain values of certain aspects of a software installation, as recited in Claims 5-10.

Moreover, as described above, comparisons of explicit values is important as a valid comparison could very likely be based on something other than mere equivalence.” (See REMARKS dated 1/31/2007, page 7)

Examiner’s Response: The argument is not persuasive, Ferchau’s teaching compares certain values of certain aspects of software installation, see Ferchau’s paragraph [0025], “The security logic verifies the integrity of the data file by comparing the retrieved software verification value with the generated software verification value.” And [0026], “The security program generates a software verification value for a data file using a software verification key maintained on the portable cryptographic device. The security program then retrieves a previously generated software verification value and verifies the integrity of the data file by comparing the retrieved software verification value with the generated software verification value.” As for the claims 5-10 features, see claims 5-10 rejections in Office action dated 11/17/2006.

Claim 11

- “Indeed, the only reasonable interpretation of this passage of Ferchau is that same agent detects the installation of a data file on the computer. When this occurs, the agent causes the Ferchau security logic to generate a verification value. As discussed above, this verification value is subsequently used in a comparison to verify the integrity of the data file. In short, this passage of Ferchau discloses that a data file is installed. Clearly, this is not ‘installation information for installing the software application onto the target computer.’” (See REMARKS dated 1/31/2007, page 8)

Examiner’s Response: The argument is not persuasive, see Ferchau’s paragraph [0009], “Another method of checking software integrity involves the use of checksums. A checksum is a type of integrity assessment code that is based on the number of set bits in the software program.... A checksum can be calculated when a software program is first installed (e.g., when a software program is known to be ‘good’), and stored, along with the checksum algorithm, on the computing device. Thereafter, the checksum can be periodically calculated, for example, before executing the software program, and compared to the previously stored checksum to ensure integrity of the software.” – Ferchau uses software’s checksum to verify the software installation, the checksum can be considered as part of the installation information for installing the software application onto the target computer.

Claims 13-48

- “Claims 13-48 were rejected for the same reasons as described above in regard to Claims 1-12. Accordingly, for the reasons discussed above, applicants assert that Claims 13-48 are in condition for allowance and request that the 35 U.S.C. § 102(b) rejections be withdrawn, and the claims allowed.” (See REMARKS dated 1/31/2007, page 8).

Art Unit: 2191

Examiner's Response: See Examiner's Responses above.

Claim 49

- “Assuming the rationale of Claim 1, Ferchau purportedly discloses a list of files stored on a portable cryptographic device which the Office Action equates, functionally, to a validation manifest. However, nothing in Ferchau discloses ‘identifying a plurality of software applications installed on the target computer,’ as recited in Claim 49, and the Office Action fails to cite where in Ferchau there is a disclosure of identifying the software applications installed on the target computer. Ferchau, at most, discloses that the device includes a list of data files.” (See REMARKS dated 1/31/2007, page 9)

Examiner's Response: The argument is not persuasive, see Ferchau's paragraph [0048], “In another embodiment, portable cryptographic device 102 may include a list of one or more data files that is to receive the requisite validation processing as described herein.” -- identifying a plurality of software application installed on the target computer.

6. Examiner is maintaining the 35 USC 102 Rejections.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 1-49 are rejected under 35 U.S.C. 102(b) as being anticipated by US 2003/0009687 A1, by Ferchau et al., hereinafter “Ferchau”.

As Per claim 1, Ferchau discloses:

- *A computer-implemented method for determining whether a software application is properly installed on target computer, comprising: obtaining a validation manifest associated with the software application, the validation manifest comprising validation actions for determining whether the software application is properly installed on the target computer;*

Ferchau's disclosure is a computer-implemented method/system for facilitating the integrity validation of data files, see Ferchau's paragraph [0093], "the present invention in at least one embodiment allows for the **validation of the integrity of data files loaded on a computing device.***(determining whether a software application is properly installed on target computer)*". Ferchau's disclosure also teaches obtaining a validation manifest associated with the software application, see Ferchau's paragraph [0048], "portable cryptographic device 102 may include a **list of one or more data files (validation manifest)** that is to receive the **requisite validation processing** as described herein"; Also see Ferchau's paragraph [0080], "Security program 106 uses the software verification key to generate a software verification value for the data file identified as needing the software verification validation (prior step 402). At step 412, security program 106 retrieves the previously generated and stored software verification value. *(validation actions)*"

- *executing the validation actions in the validation manifest; and based on the results of the executed validation actions, determining whether the software application is properly installed on the target computer.*

See Ferchau's paragraph [0093], "A security program **executes on the computing device** *(executing the validation actions)* and identifies a data file requiring integrity validation *(validation manifest)*. The security program generates a software verification value using the software verification key provided on the portable cryptographic device and uses the software verification value to **validate**

the integrity of the data file. Thus, the user can be assured that the data files loaded or **executing** on his or her computing device have not been tampered with or altered without the user's knowledge." Further in paragraph [0094], "The security program then **compares the two software verification values** to ensure that the data file has not been altered or modified. **The data file notifies the user if the two values do not match.** Thus, the user is **alerted and notified (returns the results)** of any modifications to the data files on his or her computing device." -- *determining whether the software application is properly installed on the target compute.*

As Per claim 2, Ferchau discloses:

- *The method of claim 1, wherein the validation actions comprise a validation program associated with the software application that, when executed, returns results indicating whether aspects of the software application are properly installed on the target computer.*

See claim 1 rejection, wherein the security program has same function as the validation program. Also see Ferchau's Abstract, "A method for facilitating the integrity validation of data files includes providing a portable cryptographic device having a **software verification key**, and coupling the portable cryptographic device to a computing device. The method also includes **identifying a target data file for validation** on the computing device, and **generating a software verification value for the target data file using the software verification key.** An apparatus for facilitating the integrity validation of data files includes a **software verification key** that is provided on a portable cryptographic device. The apparatus also includes a security logic that is coupled to the software verification key".

As Per claim 3, Ferchau discloses:

- *The method of claim 1, wherein the validation actions comprise a validation routine in a loadable module associated with the software application that, when called, returns results indicating whether aspects of the software application are properly installed on the target computer.*

For claim 1 feature see claim 1 rejection, for rest of claim 3 feature see Ferchau's 'security program', which is a loadable module when executed can returns results indicating whether aspects of the software application, see paragraph [0024], "a security program executes on a computing device (*loadable and executable*) and generates a software verification value for a data file loaded on the computing device." Also see Ferchau's paragraph [0011], "The present disclosure is directed to an apparatus and corresponding methods that detect unauthorized changes (e.g., modifications, substitutions, additions, deletions, etc.) to a data files, including software programs, such as, by way of example, **operating system software and application software**. The apparatus and methods may alert the authorized device user or administrator of unauthorized or improper changes to data files used in **conjunction** with computing devices, appliances, or other devices that utilize one or more data files." and [0022], "A portable cryptographic apparatus and corresponding methods, according to one embodiment, **facilitates the detection of unauthorized or improper changes to data files, including software** (e.g., operating system software, **application software**, etc.), used in **conjunction with computing devices**, appliances, or other devices that require or use one or more data files."

As Per claim 4, Ferchau discloses:

- *The method of claim 1, wherein the validation actions comprise a*

comparison instruction to compare an aspect of the software application to corresponding validation response information in the validation manifest.

For claim 1 feature see claim 1 rejection, for rest of claim 4 feature see Ferchau's Fig. 4 and description in paragraph [0081], "At step 414, security program 106 compares the previously generated software verification value and the just-created software verification value, and determines if the two values match (*comparison instruction*) at step 416." Also see paragraph [0025], "In one embodiment, a portable cryptographic device includes security logic that verifies the integrity of a data file loaded on a computing device. ... The security logic then retrieves a previously generated software verification value from, for example, the computing device, portable cryptographic device, or a remote storage device (e.g., a networked server, etc.). The security logic verifies the integrity of the data file by comparing the retrieved software verification value (*expected validation manifest data*) with the generated software verification value (*generated validation data*)."

As Per claim 5, Ferchau discloses:

- *The method of claim 4, wherein the aspect of the software application compared by the comparison instruction is the modification date of a file provided as part of the software application.*

For claim 4 feature see claim 4 rejection, for rest of claim 5 feature see Ferchau's paragraph [0091], "In one embodiment, in addition to the filename and location data, the software verification value file may also store data on the data file such as size and date to further protect against unwanted and undesirable attacks to the data file. This additional information may be used to generate the software verification value."

As Per claim 6, Ferchau discloses:

- *The method of claim 4, wherein the aspect of the software application compared by the comparison instruction is the file size of a file provided as part of software application.*

For claim 4 feature see claim 4 rejection, for rest of claim 6 feature see claim 5 rejection.

As Per claim 7, Ferchau discloses:

- *The method of claim 4, wherein the aspect of the software application compared by the comparison instruction is the version number of a shared library module used by the software application.*

For claim 4 feature see claim 4 rejection, 'version number' can be part of **filename** (see claim 4 rejection) of any software program (wherein software program can be a shared library module or a library module used by the software application), e.g. Windows 98, Windows 2000, AOL 8.0, AOL 9.0...etc., comparing version numbers is a well-known for skilled people in the art when differentiating two sets of application programs. Another way of checking version consistency is by checking the 'checksum' of application programs. See Ferchau's paragraph [0009], "Another method of checking software integrity involves the use of **checksums**. A checksum is a type of integrity assessment code that is based on the number of set bits in the software program." – if two files are in different versions, any minor change would cause the bit sets differently from the previous version, they will have different checksums. As to the shared library module, see Ferchau's paragraph [0029], "The program logic may advantageously be implemented as **one or more modules**. The modules may advantageously be configured to **reside on the computer memory and execute on the one or more processors.**"

As Per claim 8, Ferchau discloses:

- *The method of claim 4, wherein the aspect of the software application compared by the comparison instruction is the version number of a library module provided as part of the software application.*

See claim 7 rejection.

As Per claim 9, Ferchau discloses:

- *The method of claim 4, wherein the aspect of the software application compared by the comparison instruction is a system registry associated with the software application.*

For claim 4 feature see claim 4 rejection, for rest of claim 9 feature see Ferchau's paragraph [0061], "The **serial number** (a *serial number is a system registry*) is a sequence of characters that **uniquely identifies** portable cryptographic device 102. Each portable cryptographic device 102 will have a **unique serial number** (*system registry*), and no two portable cryptographic devices 102 will have the same serial numbers".

As Per claim 10, Ferchau discloses:

- *The method of claim 4, wherein the aspect of the software application compared by the comparison instruction is a system environment setting.*

For claim 4 feature see claim 4 rejection, for rest of claim 10 feature see Ferchau's paragraph [0061], "The **computing device interface type** identifies the type of portable cryptographic device 102. This information may be used to determine the type of connection 108 between portable cryptographic device 102 and computing device 104. For example, the computing device interface type may identify portable cryptographic device 102 as a USB connected module, a smart card, a

magnetic stripped card, or other type of portable storage device.” – the ‘interface type’ can be part of a system environment setting.

As Per claim 11, Ferchau discloses:

- *The method of claim 1, wherein the validation manifest further comprises installation information for installing the software application on the target computer.*

For claim 1 feature see claim 1 rejection, for rest of claim 11 feature see Ferchau’s claim 18, “the software verification value is generated in response to detecting an install of the data file (*installing software application*)”. Also see Ferchau’s paragraph [0009], “Another method of checking software integrity involves the use of checksums. A checksum is a type of integrity assessment code that is based on the number of set bits in the software program.... **A checksum can be calculated when a software program is first installed** (e.g., when a software program is known to be ‘good’), and stored, along with the checksum algorithm, on the computing device. Thereafter, the checksum can be periodically calculated, for example, before executing the software program, and compared to the previously stored checksum to ensure integrity of the software.”

As Per claim 12, Ferchau discloses:

- *The method of claim 1 further comprising, upon detecting a negative result from executing a validation action, executing a corrective action associated with the validation action.*

For claim 1 feature see claim 1 rejection, for rest of claim 12 feature see Ferchau’s Paragraph [0081], “If the two values do not match, security program 106 alerts the user, for example, by displaying a message, of the integrity violation at step 420 and ends at step 424. In one embodiment, security program 106 may provide the

user an option to proceed with the data file execution. In another embodiment, security program 106 may perform additional actions, such as, by way of example, disable operation of portable cryptographic device 102, disable operation of computing device 104, disable execution of the data file, and the like. (corrective actions)” .

As Per claim 13, Ferchau discloses:

- *A system for validating whether a software component is properly installed on a target computer, the system comprising: a processor; and a memory, the memory storing and software application, and further storing a validation module, wherein the validation module: obtains a validation manifest associated with the software application, the validation manifest comprising at least one validation action for determining whether the software application is properly installed on the target computer; executes the validation actions in the validation manifest; and based on the results of the executed validation actions, determines whether the software application is properly installed on the target computer.*

Ferchau's teaching also applies for a system with processor, memory (see Ferchau's paragraph [0028]); claim 13 is a system version of claim 1, it is rejected on the same basis as claim 1.

As Per claim 14, Ferchau discloses:

- *The system of claim 13, wherein the at least one validation action comprises a validation program associated with the software application that, when executed, returns results indicating whether aspects of the software application are properly installed on the target computer.*

For claim 13 feature see claim 13 rejection, for rest of claim 14 feature see claim 2

rejection.

As Per claim 15, Ferchau discloses:

- *The system of claim 13, wherein the at least one validation action comprises a validation routine in a loadable library associated with the software application that, when called, returns results indicating whether aspects of the software application are properly installed on the target computer.*

For claim 13 feature see claim 13 rejection, for rest of claim 15 feature see claim 3 rejection.

As Per claim 16, Ferchau discloses:

- *The system of claim 13, wherein the at least one validation action comprises a comparison instruction to compare an aspect of the software application to corresponding validation response information in the validation manifest.*

For claim 13 feature see claim 13 rejection, for rest of claim 16 feature see claim 4 rejection.

As Per claim 17, Ferchau discloses:

- *The system of claim 16, wherein the aspect of the software application compared by the comparison instruction is the modification date of a file provided as part of the software application.*

For claim 16 feature see claim 16 rejection, for rest of claim 17 feature see claim 5 rejection.

As Per claim 18, Ferchau discloses:

- *The system of claim 16, wherein the aspect of the software application compared by the comparison instruction is the file size of a file provided as part of software application.*

Art Unit: 2191

For claim 16 feature see claim 16 rejection, for rest of claim 18 feature see claim 6 rejection.

As Per claim 19, Ferchau discloses:

- *The system of claim 16, wherein the aspect of the software application compared by the comparison instruction is the version number of a shared library module used by the software application.*

For claim 16 feature see claim 16 rejection, for rest of claim 19 feature see claim 7 rejection.

As Per claim 20, Ferchau discloses:

- *The system of claim 16, wherein the aspect of the software application compared by the comparison instruction is the modification date of a file provided as part of the software application.*

For claim 16 feature see claim 16 rejection, for rest of claim 20 feature see claim 8 rejection.

As Per claim 21, Ferchau discloses:

- *The system of claim 16, wherein the aspect of the software application compared by the comparison instruction is a system registry associated with the software application.*

For claim 16 feature see claim 16 rejection, for rest of claim 21 feature see claim 9 rejection.

As Per claim 22, Ferchau discloses:

- *The system of claim 16, wherein the aspect of the software application compared by the comparison instruction is a system environment setting.*

For claim 16 feature see claim 16 rejection, for rest of claim 22 feature see claim

Art Unit: 2191

10 rejection.

As Per claim 23, Ferchau discloses:

- *The system of claim 13, wherein the validation manifest further comprises installation information for installing the software application on the target computer.*

For claim 13 feature see claim 13 rejection, for rest of claim 23 feature see claim 11 rejection.

As Per claim 24, Ferchau discloses:

- *The system of claim 13, wherein the validation module, upon detecting a negative result from executing a validation action, executes a corrective action associated with the validation action.*

For claim 13 feature see claim 13 rejection, for rest of claim 24 feature see claim 12 rejection.

As Per claim 25, Ferchau discloses:

- *A networked computing environment for validating whether a software application is properly installed on a client computer, the system comprising: a client computer upon which the software application is installed; and an administrator computer, the administrator computer operable to: obtain a validation manifest relating to the software application, the validation manifest comprising validation actions for determining whether the software application is properly installed on the client computer; carry out the validation actions in the validation manifest; and based on the results of carrying out the validation actions, determine whether the software application is properly installed on the client computer.*

Art Unit: 2191

Ferchau's teaching also applies for a networked computing environment, see Ferchau's paragraph [0006], "The modified software can transmit this compromised information to another computer over a network, where it is subsequently used to conduct unauthorized business transactions"; also see Ferchau's paragraph [0028]; claim 25 is a networked computing environment version of claim 1, it is rejected on the same basis as claim 1.

As Per claim 26 , Ferchau discloses:

- *The networked computing environment of claim 25, wherein the validation actions comprise a validation program associated with the software application which, when executed, returns results indicating whether aspects of the software application are properly installed on the client computer.*

For claim 25 feature see claim 25 rejection, for rest of claim 26 feature see claim 2 rejection.

As Per claim 27, Ferchau discloses:

- *The networked computing environment of claim 25, wherein the validation actions comprise a validation routine in a loadable library on the client computer associated with the software application which, when called, returns results indicating whether aspects of the software application are properly installed on the client computer.*

For claim 25 feature see claim 25 rejection, for rest of claim 27 feature see claim 3 rejection.

As Per claim 28, Ferchau discloses:

- *The networked computing environment of claim 25, wherein the validation actions comprise a comparison instruction to compare an aspect of the software*

application installed on the client computer to corresponding validation response information in the validation manifest.

For claim 25 feature see claim 25 rejection, for rest of claim 28 feature see claim 4 rejection.

As Per claim 29, Ferchau discloses:

- ***The networked computing environment of claim 28, wherein the aspect of the software application compared by the comparison instruction is the modification date of a file on the client computer installed as part of the software application.***

For claim 28 feature see claim 28 rejection, for rest of claim 29 feature see claim 5 rejection.

As Per claim 30, Ferchau discloses:

- ***The networked computing environment of claim 28, wherein the aspect of the software application compared by the comparison instruction is the file size of a file installed as part of software application.***

For claim 28 feature see claim 28 rejection, for rest of claim 30 feature see claim 6 rejection.

As Per claim 31, Ferchau discloses:

- ***The networked computing environment of claim 28, wherein the aspect of the software application compared by the comparison instruction is the version number of a shared library module used by the software application.***

For claim 28 feature see claim 28 rejection, for rest of claim 31 feature see claim 7 rejection.

As Per claim 32, Ferchau discloses:

- *The networked computing environment of claim 28, wherein the aspect of the software application compared by the comparison instruction is the version number of a library module installed as part of the software application.*

For claim 28 feature see claim 28 rejection, for rest of claim 32 feature see claim 8 rejection.

As Per claim 33, Ferchau discloses:

- *The networked computing environment of claim 28, wherein the aspect of the software application compared by the comparison instruction is a system registry on the client computer associated with the software application.*

For claim 28 feature see claim 28 rejection, for rest of claim 33 feature see claim 9 rejection.

As Per claim 34, Ferchau discloses:

- *The networked computing environment of claim 28, wherein the aspect of the software application compared by the comparison instruction is an system environment setting on the client computer.*

For claim 28 feature see claim 28 rejection, for rest of claim 34 feature see claim 10 rejection.

As Per claim 35, Ferchau discloses:

- *The networked computing environment of claim 25, wherein the validation manifest further comprises installation information for installing the software application on the client computer.*

For claim 25 feature see claim 25 rejection, for rest of claim 35 feature see claim 11 rejection.

As Per claim 36, Ferchau discloses:

- *The networked computing environment of claim 25, wherein the administrator computer is further operable to, upon detecting a negative result from executing a validation action, execute a corrective action associated with the validation action.*

For claim 25 feature see claim 25 rejection, for rest of claim 36 feature see claim 12 rejection.

As Per claim 37, Ferchau discloses:

- *A computer-readable medium having computer-readable instructions which, when executed, carry out the method comprising: obtaining a validation manifest associated with the software application, the validation manifest comprising validation actions for determining whether the software application is properly installed on the target computer; executing the validation actions in the validation manifest; and based on the results of the executed validation actions, determining whether the software application is properly installed on the target computer.*

Ferchau's teaching also applies for a computer-readable medium, see Ferchau's paragraph [0015], "In still another embodiment, a computer-readable storage medium has stored thereon computer instructions that, when executed by a computing device, cause the computing device to: detect a status change in a data file, the data file being loaded on a computing device; request a software verification key, ..."; also see Ferchau's claim 32. Claim 37 is a computer-readable medium version of claim 1, it is rejected on the same basis as claim 1.

As Per claim 38, Ferchau discloses:

- *The method of claim 37, wherein the validation actions comprise a validation program associated with the software application that, when executed, returns*

results indicating whether aspects of the software application are properly installed on the target computer.

For claim 37 feature see claim 37 rejection, for rest of claim 38 feature see claim 2 rejection.

As Per claim 39, Ferchau discloses:

- ***The method of claim 37, wherein the validation actions comprise a validation routine in a loadable module associated with the software application that, when called, returns results indicating whether aspects of the software application are properly installed on the target computer.***

For claim 37 feature see claim 37 rejection, for rest of claim 39 feature see claim 3 rejection.

As Per claim 40, Ferchau discloses:

- ***The method of claim 37, wherein the validation actions comprise a comparison instruction to compare an aspect of the software application to corresponding validation response information in the validation manifest.***

For claim 37 feature see claim 37 rejection, for rest of claim 40 feature see claim 4 rejection.

As Per claim 41, Ferchau discloses:

- ***The method of claim 40, wherein the aspect of the software application compared by the comparison instruction is the modification date of a file provided as part of the software application.***

For claim 40 feature see claim 40 rejection, for rest of claim 41 feature see claim 5 rejection.

As Per claim 42, Ferchau discloses:

- *The method of claim 40, wherein the aspect of the software application compared by the comparison instruction is the file size of a file provided as part of software application.*

For claim 40 feature see claim 40 rejection, for rest of claim 42 feature see claim 6 rejection.

As Per claim 43, Ferchau discloses:

- *The method of claim 40, wherein the aspect of the software application compared by the comparison instruction is the version number of a shared library module used by the software application.*

For claim 40 feature see claim 40 rejection, for rest of claim 43 feature see claim 7 rejection.

As Per claim 44, Ferchau discloses:

- *The method of claim 40, wherein the aspect of the software application compared by the comparison instruction is the version number of a library module provided as part of the software application.*

For claim 40 feature see claim 40 rejection, for rest of claim 44 feature see claim 8 rejection.

As Per claim 45, Ferchau discloses:

- *The method of claim 40, wherein the aspect of the software application compared by the comparison instruction is a system registry associated with the software application.*

For claim 40 feature see claim 40 rejection, for rest of claim 45 feature see claim 9 rejection.

As Per claim 46, Ferchau discloses:

- *The method of claim 40, wherein the aspect of the software application compared by the comparison instruction is a system environment setting.*

For claim 40 feature see claim 40 rejection, for rest of claim 46 feature see claim 10 rejection.

As Per claim 47, Ferchau discloses:

- *The method of claim 37, wherein the validation manifest further comprises installation information for installing the software application on the target computer.*

For claim 37 feature see claim 37 rejection, for rest of claim 47 feature see claim 11 rejection.

As Per claim 48, Ferchau discloses:

- *The method of claim 37 further comprising, upon detecting a negative result from executing a validation action, executing a corrective action associated with the validation action.*

For claim 37 feature see claim 37 rejection, for rest of claim 48 feature see claim 12 rejection.

As Per claim 49, Ferchau discloses:

- *A computer implemented method for determining whether a plurality of software applications are properly installed on a target computer, the method comprising: identifying a plurality of software applications installed on the target computer; and for each identified software application: obtaining a validation manifest associated with the software application, the validation*

manifest comprising validation actions for determining whether the software application is properly installed on the target computer; executing the validation actions in the validation manifest; and based on the results of the executed validation actions, determining whether the software application is properly installed on the target computer.

Ferchau's teaching also applies for a plurality of software applications, see Ferchau's paragraph [0048], "In another embodiment, portable cryptographic device 102 may include a list of one or more data files that is to receive the requisite validation processing as described herein. The list of data files is included in portable cryptographic device 102 at the time portable cryptographic device 102 is generated or made." Claim 49 is a computer implemented method for a plurality of software applications version of claim 1, it is rejected on the same basis as claim 1.

Conclusion

9. The following summarizes the status of the claims:

35 USC § 102 rejection: Claims 1-49

10. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the

Art Unit: 2191

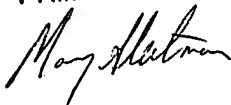
advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 8:30am - 5:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MARY STEELMAN
PRIMARY EXAMINER



Chih-Ching Chow
Examiner
Art Unit 2191
July 31, 2007

CC